**Question 9. PCA**

Both version of the questions have the following answers:

(a) 1.6

(b) Either of the following is correct:
X_{1,1} + … + X_{n,1} = 0 and X_{1,2} + … + X_{n,2} = 0
Each column of X sums to 0 / X has column-sum 0
The sum of all rows of X is 0

(c) 2, 3, 4, …

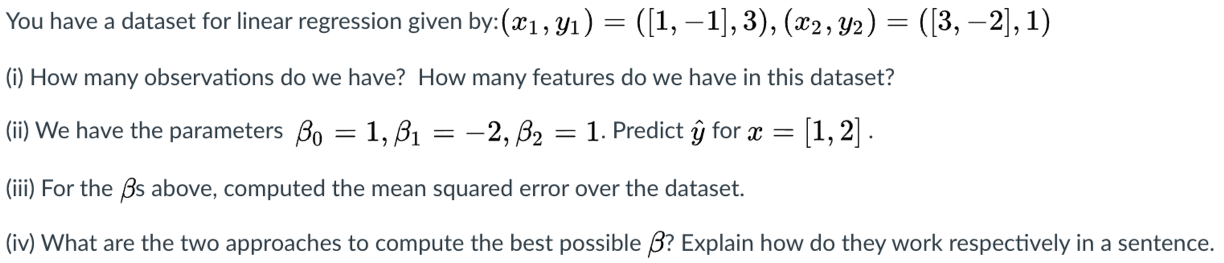
(d) Either of the following is correct:
[-0.6, 0.8]
[0.6, -0.8]

(e) Same as (d)

(f) 0.2

**Question 10. Linear Regression**

Version 1 of the question:



You have a dataset for linear regression given by: $(x_1, y_1) = ([1, -1], 3), (x_2, y_2) = ([3, -2], 1)$

(i) How many observations do we have? How many features do we have in this dataset?

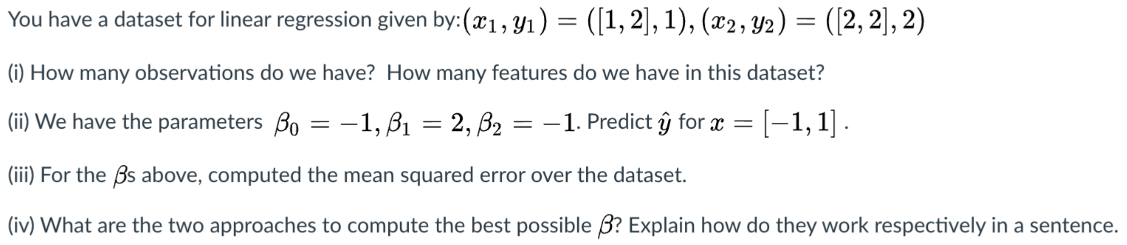
(ii) We have the parameters $\beta_0 = 1, \beta_1 = -2, \beta_2 = 1$. Predict $\hat{y}$ for $x = [1, 2]$ .

(iii) For the $\beta$s above, computed the mean squared error over the dataset.

(iv) What are the two approaches to compute the best possible $\beta$? Explain how do they work respectively in a sentence.

Answer:
(i): 2; 2
(ii): 1 + -2 * 1 + 1 * 2 = 1
(iii): MSE = 1/2 * ((1 + -2 * 1 + 1 * -1 - 3) ^ 2 + (1 + -2 * 3 + 1 * -2 - 1) ^ 2) = 1/2 * (25 + 64) = 44.5
(iv): (iterative) gradient descent and the closed-form solution/formula. Gradient descent in each step computes the partial derivatives of the MSE with respect to each beta and updates betas using the gradient, until the values of betas are converged. The closed form solution computes betas directly by minimizing the MSE.

Version 2 of the question:



You have a dataset for linear regression given by: $(x_1, y_1) = ([1, 2], 1), (x_2, y_2) = ([2, 2], 2)$

(i) How many observations do we have? How many features do we have in this dataset?

(ii) We have the parameters $\beta_0 = -1, \beta_1 = 2, \beta_2 = -1$. Predict $\hat{y}$ for $x = [-1, 1]$ .

(iii) For the $\beta$s above, computed the mean squared error over the dataset.

(iv) What are the two approaches to compute the best possible $\beta$? Explain how do they work respectively in a sentence.

Answer:
(i): 2; 2
(ii): -1 + 2 * -1 + -1 * 1 = -4
(iii): MSE = 1/2 * ((-1 + 2 * 1 + -1 * 2 - 1) ^ 2 + (-1 + 2 * 2 + -1 * 2 - 2) ^ 2) = 1/2 * 5 = 2.5

(iv): (iterative) gradient descent and the closed-form solution/formula. Gradient descent in each step computes the partial derivatives of the MSE with respect to each beta and updates betas using the gradient, until the values of betas are converged. The closed form solution computes betas directly by minimizing the MSE.

---

**Question 11. Naïve Bayes**

i. Posterior = Likelihood * Prior

P(classification | data) = P(data|classification) P(classification) / P(data)

P(classification | data) = P(data|classification) P(classification)

The Naive Bayes assumption is that all features are independent of each other conditioned on the label/classification ie. Conditional independence of features.

Explanation: these are definitions given in lecture. Note that the Naive Bayes assumption is conditional independence, not independence of features which is a much stronger assumption.

ii. Cannot classify this as an apple because the data does not contain an orange apple. Can classify this as an orange because the data contains an orange orange, a sour orange, and hard orange.

Explanation:

P(class=apple|color=orange, taste=sour, texture=hard) = P(color=orange|class=apple) P( taste=sour|class=apple) P( texture=hard|class=apple) P(class=apple)

In the dataset, P(color=orange|class=apple) = 0. Since we are multiplying values, we can see that P(class=apple|data) = 0. ie. There is 0 probability of this fruit being an apple.

P(class=orange|color=orange, taste=sour, texture=hard) = P(color=orange|class=orange) P( taste=sour|class=orange) P( texture=hard|class=orange) P(class=orange)

P(color=orange|class=orange) > 0, P( taste=sour|class=orange) > 0, P( texture=hard|class=orange) > 0, and P(class=orange) > 0. So, P(class=orange|data) > 0 ie. There is some probability of this fruit being an orange.

iii. Prior: 1/2 for both

Apple:

Likelihood: 1/6 * 3/6 * 4/6 = 1/18

Posterior: 1/2 * 1/18 = 1/36

Orange:

Likelihood: 4/6 * 4/6 * 2/6 = 4/27

Posterior: 4/27 * 1/2 = 2/27

Explanation:

Priors: additive smoothing is defined as P(class = c) = (# data points with label c + 1) / (# data points + 2) because we add 1 to each class and we have 2 classes. P(apple) = (4+1)/(8+2) = 5/10 = 1/2. P(orange) = (4+1)/(8+2) = 5/10 = 1/2

Conditional Likelihoods:

We need to do additive smoothing for each of the conditional likelihoods: P(color|class), P(taste|class), and P(texture|class).

For P(feature=f|class=c), additive smoothing is (# data points with class c where feature = f + 1)/(# data points with class c + #possible values for feature). In our case, each feature is binary, so we would add 2 in the denominator.

Computing for apple:

P(color=orange|class=apple) = (0+1)/(4+2) = 1/6

P(taste=sour|class=apple) = (2+1)/(4+2) = 3/6 = 1/2

P(texture=hard|class=apple) = (3+1)/(4+2) = 4/6 = 2/3

P(data|apple) = P(color=orange|class=apple) P(taste=sour|class=apple) P(texture=hard|class=apple) = 1/6 * 1/2 * 2/3 = 1/18

This last step is due to the naive assumption of conditional independence.

Computing for orange:

P(color=orange|class=orange) = (3+1)/(4+2) = 4/6 = 2/3

P(taste=sour|class=orange) = (3+1)/(4+2) = 4/6 = 2/3

P(texture=hard|class=orange) = (1+1)/(4+2) = 2/6 = 1/3

P(data|orange) = P(color=orange|class=orange) P(taste=sour|class=orange) P(texture=hard|class=orange) = 2/3 * 2/3 * 1/3 = 4/27

Posterior:

P(data|class=apple) P(class = apple) = 1/18 * 1/2 = 1/36

P(data|class=orange) P(class=orange) = 4/27 * 1/2 = 2/27

We still need to divide by P(data):

P(data) = 1/36 + 2/27 = 3/108 + 8/108 = 11/108

P(class=apple|data) = P(data|class=apple) P(class = apple) / P(data) = 1/36 / (11/108) = 1/36 * 108/11 = 3/11

P(class=orange|data) = P(data|class=orange) P(class=orange) / P(data) = 2/27 (11/108) = 2/27 * 108/11 = 8/11

Note 1: P(data) != P(color=orange) P(taste=sour) P(texture=hard) because the naive assumption is conditional independence of features, not independence.

Note 2: In this case, all features are binary and there are equal amounts of data in both classes, so the answer may be the same if we add an incorrect value to the denominators. But this may not hold when features have different number of categories or different amount of data per category.

Note 3: also possible to find prior and posterior for apple and find values for orange by doing 1 - prior(apple) and 1-posterior(apple)

iv. Predict Orange

Explanation:

From part iii, P(orange|data) > P(apple|data), so we would classify this new fruit as an orange.

v. Use log probabilities to prevent underflow (the result of multiplying probabilities requires more precision than the computer can store)

Explanation: this was discussed in class and in homework 2

**Question 12. Clustering**
**Version 1 Question.**

Consider a set of 2-D points located at:

(-2, 1), (1, 2.5), (0, 0.5), (-1, 0), (3, 2.5)

Use Euclidean distance for distance computation.

(i) Consider 2-means clustering with 2 initial centers at (0, 0.5) and (-1, 0). Perform two iterations of the algorithm (write down your initial assignments, the new centers, and the next assignments).

(ii) Perform single linkage clustering and write down your tree.

(iii) Perform complete linkage clustering and write down your tree.

(iv) Create and write down a 4 point dataset (also 2-D) where single linkage and complete linkage produce the same tree.

**Answer**:

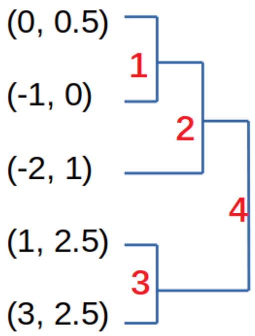
(i) iteration 1:

center 1: (0, 0.5). center 2: (-1, 0)
cluster 1: (0, 0.5), (1, 2.5), (3, 2.5)
cluster 2: (-1, 0), (-2, 1)

iteration 2:

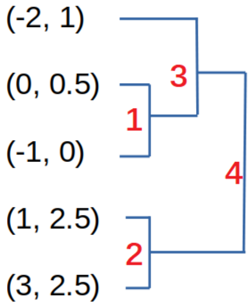
center 1: (4/3, 11/6). center 2: (-1.5, 0.5)
cluster 1: (1, 2.5), (3, 2.5)
cluster 2: (-2, 1), (-1, 0), (0, 0.5)

(ii)



(Red numbers are the order of clustering)

(iii)



(iv) (0, 0), (1, 0), (4, 0), (9, 0)

**Version 2 Question**:

Consider a set of 2-D points located at:

(-1, 3.5), (-4, 2), (-2, 1.5), (1, 3.5) ,(-3, 1)

Use Euclidean distance for distance computation.

(i) Consider 2-means clustering with 2 initial centers at (-2, 1.5) and (-3, 1). Perform two iterations of the algorithm (write down your initial assignments, the new centers, and the next assignments).

(ii) Perform single linkage clustering and write down your tree.

(iii) Perform complete linkage clustering and write down your tree.

(iv) Create and write down a 4 point dataset (also 2-D) where single linkage and complete linkage produce the same tree.

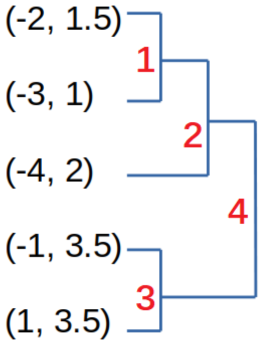
**Answer**:
(i) iteration 1:

center 1: (-2, 1.5). center 2: (-3, 1)
cluster 1: (-2, 1.5), (-1, 3.5), (1, 3.5)
cluster 2: (-4, 2), (-3, 1)

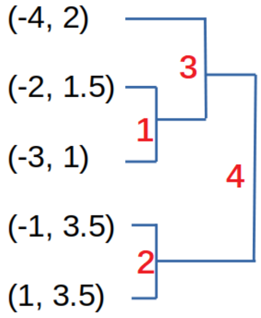
iteration 2:

center 1: (-2/3, 17/6). center 2: (-3.5, -1.5)
cluster 1: (-1, 3.5), (1, 3.5)
cluster 2: (-4, 2), (-3, 1), (-2, 1.5)

(ii)



(iii)



(iv) (0, 0), (1, 0), (4, 0), (9, 0)

**Question 13. Neural Networks**

1. First layer: (1+3)*4
Second layer: (1+4)*2
Output: (1+2)
Total: 29

2. First layer activations: (2,2,2,2)
Second layer activations: (8,8)
Output: 16

3. Choose a positive learning rate a.

```
Initialize the parameters
For t=1,2,3,…
  Randomly sample a subset from dataset, then update the parameters by
  w(t) = w(t-1)-a*(average gradient of loss function on the subset).
  Repeat until converges.
```

4. The difference is that SGD only uses a subset to update the network, but GD uses the whole dataset. The reason is that the whole dataset is too big to update the network. It costs a lot of time.